Distilled Datamodel with Reverse Gradient Matching

Supplementary Material

In this document, we present supplementary materials that couldn't be accommodated within the main manuscript due to page limitations. Specifically, we offer additional details on the proposed DDM framework, elucidating the computation of the hierarchical attribution matrix through our framework and the methodology employed for data clustering.

6. More Details of DDM Framework

6.1. Data Clustering of DDM

To enhance tracing performance, we concentrate on calculating the weighting matrix for each batch of data, rather than for each individual image. That is, the total cluster number K is set as $L < K < |\mathcal{D}|$ (L is the total number of the class). This approach is taken as the impact of a single image becomes negligible when training with a large number of images.

In our proposed DDM framework, we employ data clustering to partition the training data into several distinct batches. Specifically, we utilize a pre-trained feature extractor to embed the training images into the same feature space. Subsequently, we apply K-means clustering to the features, denoting each cluster of data as $\mathcal{D}_{c,k}$, where $l = 1, 2, ..., L$ and $c = 1, 2, ..., C$. This implies that we cluster the images within a class into C clusters. For each class of data, we set the number of clusters C to be 10.

6.2. Accelerating with Hierarchical DDM

As stated in the main paper, to enhance the efficiency of online evaluation, we employ hierarchical attribution calculation, which expedites the leave-one-out retraining process. Note that on the offline training stage, both the class-wise and cluster-wise synset are learned, where the class-wise one is denoted as $\mathcal{S}_{class} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_L\}$ and the cluster-wise one is denoted as $\mathcal{S}_{cluster} = \{\mathcal{S}_{l,1}, \mathcal{S}_{l,2}, ..., \mathcal{S}_{l,C}\}_{l=1}^{L}$ and $K = L \times C$. With this hierarchical synset, we don't need to apply the retraining to the entire perturbation set $|P_t|$. Instead, it is calculated as:

$$\begin{aligned} l &\leftarrow \arg\max_{l} W_l, \quad 1 \leq l \leq L, \\ c &\leftarrow \arg\max_{l} W_{l,c}, \quad 1 \leq c \leq C \end{aligned} \tag{11}$$

In the first equation, we require $|P_t| = L$, and in the second one, we need $|P_t| = C$. Therefore, with this hierarchical synset, we only need to retrain the networks $C + L$ times, significantly reducing the original $C \times L$ retraining to $C + L$.

In the standard setting for the CIFAR-10 dataset, where $K = 10 \times 10$, and for the CIFAR-100 dataset, where $K = 100 \times 10$, the hierarchical synset accelerates the process by 5 times in the CIFAR-10 dataset and 10 times in the CIFAR-100 dataset.

6.3. Reverse Gradient Matching for Matching Performance Enhancement

In the main paper, we claim that our proposed reverse gradient matching provides enhanced matching performance, which has been proved in the experiment ('DDM could be used as a quick unlearn method'). The reasons for it can be analyzed by the accumulated errors. Since we performance the leave-one-out retraining, which means that each time we unlearn one data cluster. Thus, for unlearning the data cluster $\mathcal{D}_\kappa$, the errors of our proposed reverse gradient matching can be denoted as:

$$\epsilon_\kappa = \sum_t \left| \nabla\mathcal{L}(\theta_{\tau-t}, \mathcal{S}_\kappa) + \nabla\mathcal{L}(\theta_t, \mathcal{D}_\kappa) \right|, \tag{12}$$

where $1 < \kappa < K$ and $\mathcal{S}_\kappa$ is corresponding synset that matches the data cluster $\mathcal{D}_\kappa$ with reverse gradients. And denote $\mathcal{X}_\kappa$ as the corresponding synset that matches the data cluster $\mathcal{D}_\kappa$ with normal gradients, then the accumulated errors could be denotes as:

$$\epsilon_\kappa = \sum_t \sum_{k \neq \kappa} \left| \nabla\mathcal{L}(\theta_t, \mathcal{X}_k) - \nabla\mathcal{L}(\theta_t, \mathcal{D}_\kappa) \right|, \tag{13}$$

which, compared with Eq. 12, introduces an additional summation operation, resulting in larger accumulated error values.

7. Experimental Setting

The experimental setting of the proposed DDM framework is depicted in Table 5, where the dataset information, network backbones (ConvNet, AlexNet, ResNet and Simple Vit) are listed. Two different distance functions $\mathcal{D}ist(\cdot)$ are utilized in Eq. 6 of the main paper.

8. More Experiments

8.1. Class-wise DDM vs Cluster-wise DDM

In our proposed DDM framework, both the class-wise and the cluster-wise synthetic images are obtained for efficient tracing by hierarchical search (details in Sec. 6.2). In Fig. 6, we depict the tracing results (got by Eq. 11) with the most influential datapoints in terms of $\mathcal{D}ist_1$, with $\|X_t\| = 1$ ('one image inference' with groundtruth label as '5') and $\|X_t\| = 256$ ('a batch of images inference' with

Table 5. The detailed settings in the experimental implementation.

Dataset	**Network**	**Settings**					
		lr_net	lr_img	Training Epochs	Step for Synset	Step Length	Dist
MNIST	ConvNet/AlexNet/ResNet	0.001	0.1	30	50	4	Cosine Dist
	Simple ViT	0.001	0.005	30	50	4	MSE
CIFAR10	ConvNet/AlexNet/ResNet	0.01	0.1	30	50	4	Cosine Dist
	Simple ViT	0.01	0.005	30	50	4	MSE
CIFAR100	ConvNet/AlexNet/ResNet	0.01	0.1	30	50	4	Cosine Dist
	Simple ViT	0.01	0.005	30	50	4	MSE
TinyImgNet	ConvNet/AlexNet/ResNet	0.01	0.1	30	50	4	Cosine Dist
	Simple ViT	0.01	0.005	30	50	4	MSE

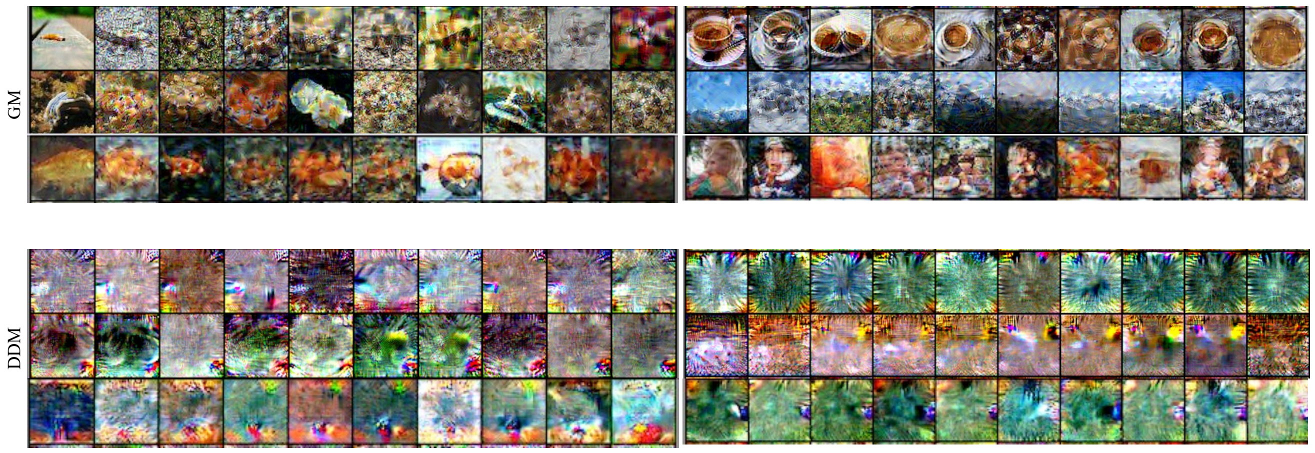

Figure 5. Visualization of condensed 10 image/class with ConvNet for TinyImageNet dataset. We compare the visualization results between gradient matching (GM) and reverse gradient matching (DDM). In each visualization, each column represents a condensation of a cluster.

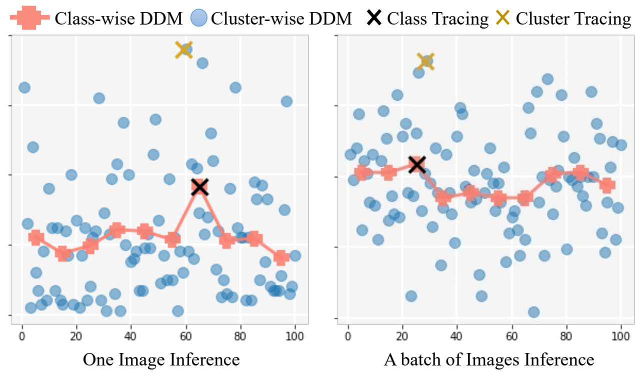

Figure 6. Locating the most influential data points, where the results are computed with distance function $Dist_1$ on CIFAR10 dataset. We firstly locate the class with the class-wise synset and then to the cluster with the cluster-wise synset of that class.

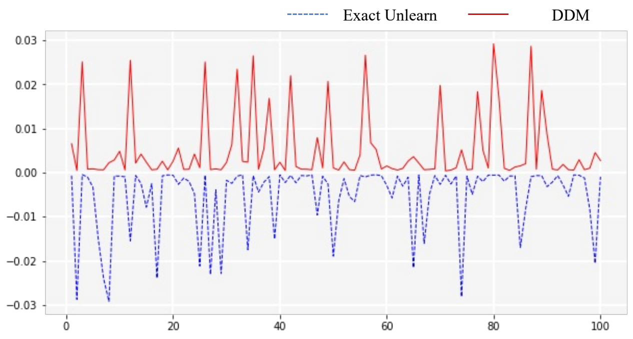

Figure 7. Comparing the empirical error between the exact unlearn model and the DDM model. We scale and flip 'exact unlearn' for better visual comparison.

all groundtruth labels as '2'). As can be observed, both the class-wise tracing and cluster-wise tracing give consistent tracing results. In addition, DDM works on both the sin-

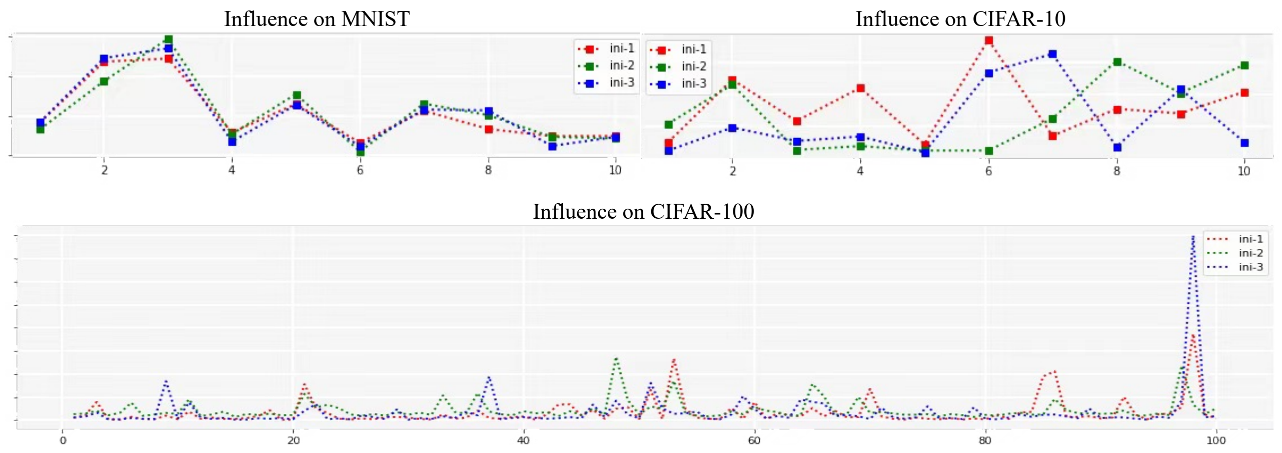

Figure 8. Comparison of the training data attribution matrix with different initializations. The experiments are conducted on MNIST, CIFAR-10 and CIFAR-100 datasets, with the ConvNet as the base network.

gle image inference (gt label belongs to the 5-th label) and multi-image inference cases (all gt labels belong to the 2-th label). indicating the robustness of the DDM framework.

8.2. More Visualization Results

In Fig. 4 of the main paper, we visualize the synset on MNIST dataset and CIFAR-100 dataset. Here, we provide additional visualization results on the TinyImageNet dataset, as depicted in Fig. 5.

The figure provides additional evidence supporting the privacy protection capabilities of our proposed DDM, as the synsets exhibit no recognizable objects. Moreover, the uniqueness of synthetic images learned from each data cluster highlights the importance of the clustering operation.

8.3. Comparing DDM for Machine Unlearning with Exact Unlearn

As discussed in the method section, we optimize the synthetic images using the proposed reverse gradient matching, which are then employed to fine-tune the target network to mitigate the impact of specific data clusters. It's important to note that we don't anticipate the DDM framework to perfectly mimic the exact unlearned model. Instead, our focus is on whether it can capture important characteristics of the model. In this experiment, we use empirical error for comparison, as depicted in Fig. 7. In the figure, we scale and flip 'exact unlearn', which, in an ideal situation, should be symmetric to 'DDM'. As observed, they are roughly symmetrical, indicating that the DDM models can serve as a surrogate for analyzing the target model and perform well in eliminating certain data points compared to the exact unlearn.

8.4. How Did Different Initializations Influence the Network?

Here, we investigate the impact of different initializations on the computed training data matrix. We perform this comparative experiment on the MNIST, CIFAR-10, and CIFAR-100 datasets trained using the ConvNet architecture. We consider three different initializations: 'Kaiming' (ini-1), 'Normal' (ini-2), and 'xavier' (ini-3).

The experimental results are illustrated in Fig. 8, revealing the following observations:

- The training data attribution is robust across different initialization methods, yielding similar attribution matrices. This observation holds true for the MNIST, CIFAR-10, and CIFAR-100 datasets.
- With an increase in the size of the training data, the attribution matrices learned from the three different initializations become more diverse. This divergence may arise from the selection of the basic ConvNet as the backbone, potentially leading to local optima.

9. DDM for Analysis of Other Model Behaviors

In the main paper, we detailed instructions on analyzing networks by identifying the most influential training data for certain test images. We emphasize that both the local and global behaviors of the network can be captured by the training data matrix when using specific distance functions defined by the certain evaluation objective.

Here, we present various distance functions for different evaluation objectives.

9.1. Inference Function of Certain Test Samples

The distance function is defined in Eq. 10 in the main paper. This function aims to identify which part of the training data is responsible for the final prediction. We provide different

Table 6. Detecting noisy training data for the target network. We add random noise with perturbation norm of 0.1 to each image.

Percentage	**0%**	**10%**	**20%**	**30%**	**40%**
Random Select	78.4	70.5	68.8	61.3	55.7
DDM	78.4	82.0	79.3	73.6	70.1

evaluation objectives, including those influencing the current predictions most/least and those contributing the most to making correct predictions.

And the corresponding experiments have been listed in the main paper in Table 1.

9.2. Model Diagnostic for Low-quality Training Samples

To identify essential part of data that contributes to the overall prediction ability. We have already displayed the corresponding experiments in Table 3 by sorting the training cluster with its influence to the final network performance. To be specific, we randomly choose part of training data as the validation set $\mathcal{V}$, then we could determine distance function for the model global performance evaluation as:

$$\sum_{x_t \in \mathcal{V}} \mathcal{D}ist_2(x_t), \tag{14}$$

where $\mathcal{D}ist_2$ is pre-defined in Eq. 10 for identifying those with the most influence on whether the model makes correct predictions.

In addition to the previous experiment focused on removing locally low-quality data from the training set, we conducted an additional experiment involving the introduction of random noise to 20% of the training data. Subsequently, we removed the data based on the evaluation objectives mentioned earlier.

The experimental results are presented in Table 6. The table reveals that the addition of random noise has a detrimental effect on the performance of the target model. However, when selectively removing data based on our proposed DDM, the network's performance improves, demonstrating a more significant improvement compared to the 'Random Select' approach.

9.3. Transferability Between Different Networks.

And we also find that our proposed DDM also provides to measure and improve the transferability between different networks. To be concrete, in the typical work like knowledge distillation, the student network can be trained by:

$$\mathcal{L}_{\text{total}} = (1-\alpha) \cdot \mathcal{L}_{ce}(\mathcal{N}(x), y^t) + \alpha \cdot \mathcal{KL}(\mathcal{N}(x), \mathcal{M}(x)), \tag{15}$$

where $\mathcal{N}$ represents the student network to be trained, $\mathcal{M}$ is the target network, $\mathcal{KL}$ denotes the KL-divergence, and α is the balancing weight. It is important to note that not all networks may experience performance improvement through such distillation, as there could be conflicts that hinder the overall performance enhancement.

Table 7. DDM for network tranferability while distillation.

Percentage	**0%**	**10%**	**20%**	**30%**	**40%**
Random Select	94.1	93.5	88.7	87.3	85.8
DDM	94.1	94.2	94.2	92.9	87.3

We use the KL-divergence as the distance function and calculate the whole evaluation distance in the similar way as in Eq. 14. The experiments are conducted on Table 7. In the table, we delete some percentages of data from the network training and distill it to train the student network. The experiments are conducted on CIFAR-10 dataset on ResNet-18, and the teacher network is optimized by knowledge undistillation. Thus, directly distill from the teacher would cause accuracy drop. And deleting some samples from the training data could attack such knowledge distillation and improve the network performance.

9.4. To be Explored.

In this paper, we distilled the training gradients to several synthetic images, which enables the fast impact elimination. Thus, it is possible to build the training data attributes with the distance function defined between networks, which shows great potential to explore other kinds of network behaviors.